Project Overview

SampleScape2k17 is a sequencer that creates a unique aural landscape using triple-thread semi-asychronous time-delay processing. At the hands of even the most amateur artiste, a profound, meaningful juxtaposition of classical and organic sounds in glorious disarray is but keystrokes away.

Results

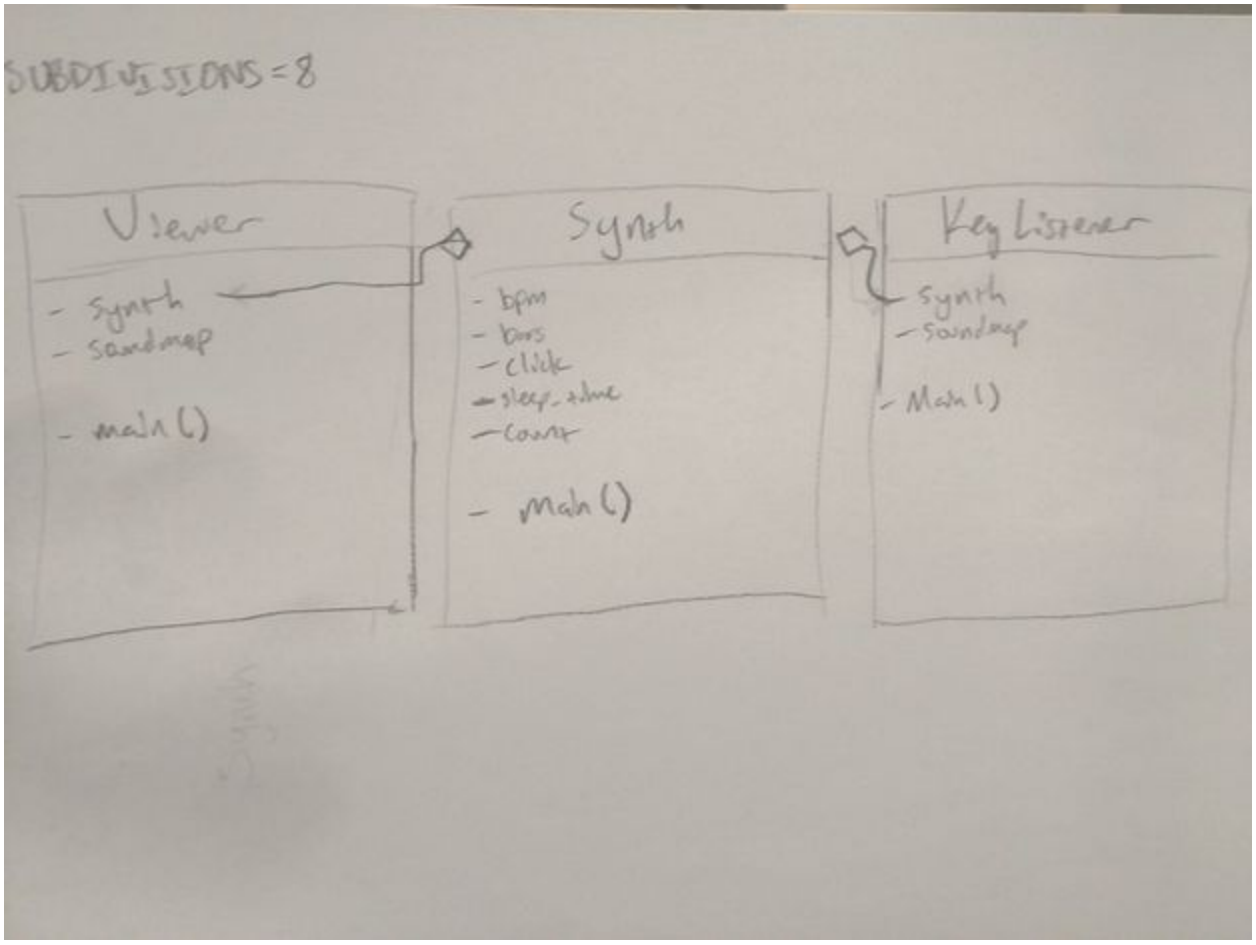
We created a sequencer using an MVC framework. It takes samples, maps them to keys and then loops on a user specified number of bars at a user specified tempo. Beats are subdivided and quantized. From here, a user can play and layer samples as the track loops. A simple demo is found here: https://youtu.be/_026DegVCxo

Unfortunately, in the current implementation, Python struggles to keep constant time. This results in rather freeform but coherent music.

Most of what we accomplished was learning how not to accomplish certain tasks, or rather the limitations of certain design decisions. While this was really valuable, it is somewhat hard to demonstrate in a deliverable. More specifically, we explored quite a few different audio frameworks, aiming at doing live audio synthesis. However, there were many roadblocks along this route. In a similar vein, we worked with multiple visual toolkits, finally settling on tkinter for a very simple input task.

We did learn a lot about threading, and more into the inner workings of the python interpreter.

Implementation



Our system is built in a classic MVC architecture. The main model class, Synth, holds onto the state information for our project, which in this case is a list of lists. This data structure represents the individual time subdivisions in our sequencer loop. Each time subdivision is itself another list, which holds onto all of the sound samples that need to be played during that step. This was a difficult design decision for us, because initially we were planning to stream audio data to a player class, and eventually we had to roll back that plan and make a discretized model instead. The model also has a method that iterates through its loop of sounds, incrementing a counter, which the controller uses to place sounds into the list. While it is iterating through the list, it stores any sounds in the list to the 'play queue', which the view uses to play sounds.

The other two parts of our system are the View and the Controller, Viewer and KeyListener. KeyListener and Viewer each read in the names of the provided audio samples from a text file, creating mappings from keys to sound tags or sound tags to sounds respectively. Each of them also has a main method. In KeyListener, the main method loops through the pygame event queue, allowing for keydown events to be processed and the proper sounds added to the sequencer loop, or also to propagate the quit message. In Viewer, the main method iterates through the 'play queue', mapping sound tags to sounds and playing them.

As an addition, we have a small TKinter dialog box upon initialization that captures user input for variables. This was mainly an excuse to play around with different visual frameworks and interactions.

Reflection

The scope of this project was good for what we initially knew. If we had to go back and redo it, we probably would not attempt real time audio synthesis, instead exploring more interface and visualization aspects. Our iterative design plan actually worked fairly well, mostly cycling in the exploratory testing phase, or debugging. Whenever we made design decisions, we white boarded an implementation plan, which continues to be a very effective strategy for keeping teams on track.

Although we did do a fair amount of design and exploratory development spread during the two week period, most of the final product was developed in ~7 hours of high intensity pair programming. While suboptimal in some respects, this is a very comfortable and productive method for both of us. The hackathon dynamic was planned, but planned earlier in the two week cycle. Due to the rougher than anticipated path to MVP, this was pushed back.

In the future, holding to internal deadlines can help maintain a project on track, even if it forces design changes. The learning from failures was really valuable, but it did put a dampener on all the features we wanted to implement in our project.